제1회 한국사이버테러정보전학회 춘계학술발표대회 (2004.5)

프로그램 소스코드 취약성 분석에 관한 연구

하 경 휘*, 최 진 우*, 우 종 우*, 김 홍 철**, 박 상 서**
* 국민대학교 컴퓨터학부
** 국가보안기술연구소

요 약

최근의 침해 사례를 보면 공격자들이 순수 공격 기술이나 네트워크 구성의 결함을 이용하기 보다는 시스템 상에서 구동 중인 프로그램들, 특히 서비스를 위한 프로세스들이 개발 당시에 가지는 근본적인 취약성을 이용한다. 따라서 보안 관리자들은 공격에 이용되는 취약성을 보완하기 위해 많은 노력과 시간을 투자해야만 하며, 동시에 개발자들 또한 계속된 프로그램 수정 작업으로 인한 부담이 커지고 있는 실정이다. 이러한 문제점을 해결하기 위해서는 우선적으로 소스 코드 내에 잠재되어 있는 취약한 함수들에 대한 분석이 있어야 한다. 본 논문에서는 프로그램의 표준 C 함수에 관련된 취약성과 Win32 API 함수의 취약성에 대하여 분석하고, 그 결과를 기반으로 소스 코드의 취약성을 검사하기 위한 자동화 도구를 제안한다.

A Study on the Analysis of Vulnerabilities in the Program Source Code

Ha Kyung Hui*, Choi Jin Woo*, Woo Chong Woo*, Kim Hong Chul**, Park Sang Seo**
* Kookmin University, ** National Security Research Institute

ABSTRACT

The majority of recent intrusions reveal that the attackers do not use the previous intrusion techniques or network flaw, rather they tend to use the vulnerabilities residing inside the program, which are the running programs on the system or the processes for the service. Therefore, the security managers must focus on updating the programs with lots of time and efforts. Developers also need to patch continuously to update the program, which is a lot of burden for them. In order to solve the problem, we need to understand the vulnerabilities in the program, which has been studied for some time. And also we need to analyze the functions that contains some vulnerabilities inside. In this paper, we first analyzed the vulnerabilities of the standard C library, and Win32 API functions used in various programs. And then we described the design and implementation of the automated scanning tool for writing secure source code based on the analysis.

1. 서 론

인터넷 사용의 급격한 증가와 정보 통신 기술의 급속한 발전으로 인해 보안은 상호 연결된 시스템들 사이의 안전한 접속에 관한 문제가 쟁점이 되어 왔다. 그러나 침입 사례의 대부분은 공격자의 순수 공격 기술과 네트워크 구성의 결함을 이용하기 보다는 시스템 상에서 구동 중인 프로그램들, 특히 서비스를 위한 프로세스들이 개발 당시에 가지는 근본적인 취약성들을 이용한다.

이러한 취약성들을 보완하기 위한 보안 관리 사이클(Security Management Cycle)에 있어서 시스템 보안 관리자들에 의해 수행되는 대부분의 작업들은 각각의 시스템들에서 구동 중인 프로그램들의 결함을 보완하는 것이다. 이를 위해 제공되는 보안 패치(Secure Patch)들의 설치 작업이 보안 관리 사이클의 대부분을 차지한다.

본 논문에서는 이러한 문제들을 원처적으로 해결하기 위하여 프로그램 개발 시 공격의 개연성을 가지는 취약성들에 대한 실질적인 코드들을 조사 및 분석하고 이를 위하여 소스 코드 수준에서 잠재할 수 있는 취약성을 자동으로 분석하고 이에 따른 지침서를 제공하는 자동화 도구의 개발을 제안한다[1][2][3].

2. 함수 취약성 분석

프로그램 내 잠재적인 취약성을 해결하기 위해서는 프로그램 작성 단계에서 보다 명확한 보안 개념을 가지고 소스 코드를 작성하여 취약성을 원천적으로 근절해야 한다. 이를 위해서 함수 취약성의 대표적인 부류들에 대한 정의 및 각 부류들의 잠재적인 취약성과 그 원인에 대하여 심도 있는 이해가 수반되어야 함은 명백한 사실이다. 본 장에서는 이를 위하여 잠재적인 취약성이 내재되어 있는 다양한 부류의 함수들에 대하여 기술하고[4][5][6], 이들 취약성들을 검사하는 대표적인 몇몇 도구들에 대하여 기술한다.

2.1 표준 C 프로그램에서의 취약성

잠재적인 취약성을 내포하고 있는 표준 C 함수의 취약성 함수들로 인해 초래될 수 있는 위협은 다음과 같이 4 가지 부류로 요약될 수 있다[7].

- ❑ 버퍼 오버플로우(buffer overflow) 관련 취약성
  버퍼 오버플로우는 스택 오버플로우와 힙 오버플로우를 구분되며, 고정된 버퍼 크기 이상의 데이터를 저장하려 할 경우 야기되는 문제로써 버퍼를 오버플로우 시켜 공격자가 원하는 수행을 가능하게 한다. 버퍼 오버플로우 공격은 저장할 버퍼의 크기가 적합한지를 판별하기 위한 검사 코드가 구현되지 않은 프로그램들을 목표로 삼으며, 그 결과 해당 서비스의 접근 거부, 또는 시스템의 다운 등을 초래한다. 관련 함수로는 chroot, gets, scanf, strcat 등이 있다.
- ❑ 경쟁 상태(race condition) 관련 취약성
  일반적으로 프로세스가 원자적으로 실행되지 않기 때문에 발생하는 문제이다. 프로그램의 다중 접근이 부적절할 경우 다른 프로세스에 의해 현재 프로세스가 방해를 받아 공유 자원의 불법적인 접근을 가능하게 한다. 관련 함수로는 access, chmod, exec, mktemp 등이 있다.
- ❑ 접근제어(access control) 관련 취약성
  대부분의 경우 시스템 자원 접근에 대한 "setuid/setgid bit"를 설정하는 함수들에 의해서 발생한다. 이 취약성을 이용하여 공격자는 특정 그룹 또는 사용자의 권한을 획득할 수 있다. 관련 함수로는 chmod, chcwn 등이 있다.
- ❑ 포맷 스트링(format string) 관련 취약성
  비 신뢰 데이터를 입력으로 사용하는 경우로

써 포맷 지시자를 포함한 스트링을 인자로 하는 포맷 함수 계열에서 발생할 수 있다. 공격자는 프로그램의 스택 내부를 이해하여 프로세스의 사용 메모리 내부의 원하는 위치의 값을 조작할 수도 있다. 관련 함수로는 printf, sacnf 등이 있다.

2.2. Win32 API 에서의 취약성 및 해결안

Win32 API 함수의 취약성 함수들에 대한 6 가지 부류를 알아보고, 해당 부류의 함수들에 대한 취약성 원인과 그 해결방안을 제시한다.

- ❑ 접근위반(access violation) 관련 취약성
  이 부류의 함수들은 "NULL" 문자를 포함하지 않은 문자열을 적절히 다루지 못하는 문제점을 가지고 있다. 만약 "NULL" 문자를 포함하지 않은 문자열을 다룰 경우 "Access Violation"이 발생하며 프로그램이 비정상적으로 종료될 수 있다. 이를 해결하기 위해 버퍼를 "NULL" 값으로 미리 초기화 해 줌으로 취약성을 예방할 수 있다. 관련 함수로는 _mbslen, _tcslen 등이 있다.
- ❑ 버퍼 빅(buffer big) 관련 취약성
  이 부류의 함수들은 수행할 문자열의 크기가 목적 버퍼의 크기보다 크거나 목적 버퍼의 크기와 같을 경우 버퍼 오버플로우가 발생하는 문제점을 가지고 있다. 이를 해결하기 위해서는 지정된 버퍼의 크기를 충분히 크게 해 주거나 입력값을 버퍼의 크기보다 작게 제한해 주는 방법이 있다. 관련 함수로는 _mbsnbcpy, CopyMemory, StrNCat 등이 있다.
- ❑ 포맷 스트링 (format string) 관련 취약성
  표준 C 함수의 포맷 스트링 관련 취약성과 동일한 부류이다. 관련 함수로는 _cprintf, _tprintf, _ftprintf 등이 있다.
- ❑ 임계영역(critical section) 관련 취약성
  이 부류의 함수들은 임계 영역을 다루는 기능을 수행하는 함수들이다. 일반적인 경우에는 문제가 발생하지 않지만 적은 메모리 상태일 경우 예외가 발생할 수 있다. 각 초기화 함수들에 대해서 대체함수를 사용하거나 예외 처리 구문을 적용하는 방법으로 예방할 수 있다. 관련 함수로는 EnterCriticalSEction 등이 있다.
- ❑ 프로세스 실행에 의한 취약성
  이 부류의 함수들은 환경 변수 "path"에 의존적으로 실행 파일을 로드하는 문제점을 가지고 있다. 만약 공격자가 악의적인 목적의 프로그램을 해당 실행 파일과 같은 이름으로 하여 주요 경로의 우선되는 위치에 놓아둔다면 악의적인 목적의 프로그램이 실행 될 수 있다. 이를 예방하기 위해서는 프로세스 실행을 위한 파일의 절대 경로를 명시함으로써 예방 할 수 있다. 관련 함수로는 CreateProcess, ShellExecute 등이 있다.
- ❑ 실행 프로그램 확장자 관련 취약성
  이 분류에 속하는 함수들은 파일명만 주어지거나 공백 문자가 있을 경우 오동작을 하는 문제점을 가지고 있다. 동일 디렉터리에 "COM" 파일과 "EXE" 파일이 있을 경우 "COM" 파일이 먼저 실행되며, 공백 문자의 경우 공백 문자 이전까지의 경로만 이해한다. 이런 문제점을 이용해 공격자가 악의적인 목적의 프로그램을 실행시킬 수 있다. 이를 해결하기 위해서 실행 파일들의 확장자를 명시하고 공백 문자에 대한 처리를 해주는 방법이 있다. 관련 함수로는 _execl, _execv, _spawnl, _spawnv 등이 있다.

2.3 시스템 사례

소스코드의 취약성을 검사하는 대표적인 5 가지의 도구들에 대하여 설명한다[8].

- ❑ Pscan
  C 소스코드 내에서 악용되는 버퍼 오버플로우와 포맷 스트링 공격을 발견하는 도구로써

그 기능이 한정되어 있다[9].

- Flawfinder

  Pscan 도구와 그 기능이 유사하지만 보다 많은 유형의 오류들을 발견하며, 위험 레벨(risk level)에 의해 그 결과를 정렬할 수 있다[10].

- RATS

  C, C++, Perl, PHP, 그리고 Python과 같이 여러 개발 언어들를 스캔하는 기능을 가지는 유일한 검색 도구이다. 기능은 Flawfinder와 유사하다. 실행 결과는 HTML로 생성된 보고서를 제공한다[11].

- Splint

  다른 도구들과는 달리 코딩 시 발생하는 실수들과 반드시 요구되어야 하는 구조적 코딩 스타일을 찾아낸다. 비교적 덩치가 가볍고, 보안이라는 관점에서의 기능이 빈약하다[12].

- MOPS

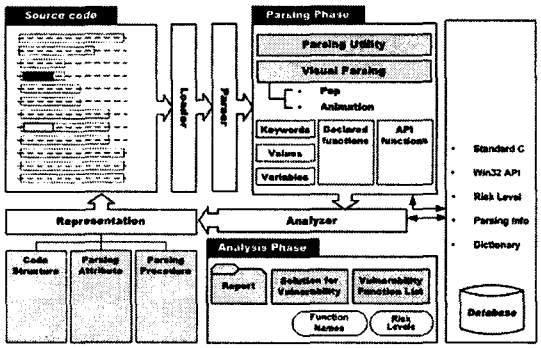
  FSM(Finite State Machines)을 사용한 모델을 생성하는 방법을 알아야만 소스코드 분석이 가능하다. 다른 도구들에 비해 사용법이 간단하지 않고, Java Run-time 환경을 요구한다[13].

3. 자동화 도구의 설계 및 구현

본 논문에서는 소스코드 분석을 통하여 취약성을 가진 표준 C 함수들과 Win32 API 함수들을 발견해 내고, 발견된 이들 함수들이 가지는 취약성을 제거하기 위한 해결방안을 제시하는 자동화 분석 도구를 제안한다. 본 장에서는 소스코드의 분석을 위한 자동화 도구의 전체적인 구조와 개별적인 모듈에 대해 설명하고 이 들의 상호 연관성에 대해서 상세하게 설명한다.

3.1 설 계

[그림 1]은 소스코드 분석을 위한 자동화 도구의 전체적인 시스템 구조를 도식화 한 것이다. 이 자동화 도구는 소스 코드 분석 작업을 위해 파싱부(Parsing Phase)와 분석부(Analysis Phase)의 2 단계로 구성되어 있다. 파싱부는 소스코드에 대한 파싱과 관련된 기능을 수행하고, 분석부는 파싱된 결과를 가지고 함수의 취약성을 분석하는 기능을 수행한다.



[그림 1] 시스템 구조

소스코드 분석 작업에 대한 전체적인 시스템 흐름은 다음과 같다.

① 소스 코드를 읽어 파싱 모듈에 의해서 시각적인 특징이 강조된 파싱 과정에 들어간다.

② 파싱 방법이 결정되면 파싱 모듈은 데이터베이스에서 파싱과 관련된 정보를 가져온다. 이 정보를 기반으로 소스 코드에 대한 파싱 과정이 수행되고, 진행되는 결과들이 소스 코드의 에디팅 영역과 부가적인 다이얼 로그 창에 적용된다.

③ 파싱 과정이 종료 된 후 사용자의 요청에 의해서 분석 과정이 수행된다.

④ 분석 과정이 시작되면 분석 모듈은 데이터베이스에 있는 취약성 함수 정보와 위험도와 관련된 정보를 가져온다. 이 정보를 기반으로 소스 코드에서 발견된 취약성 함수에 대해서 소스 코드 에디트 영역에 위험도에 따라 다른 색상으로 표기 한다.

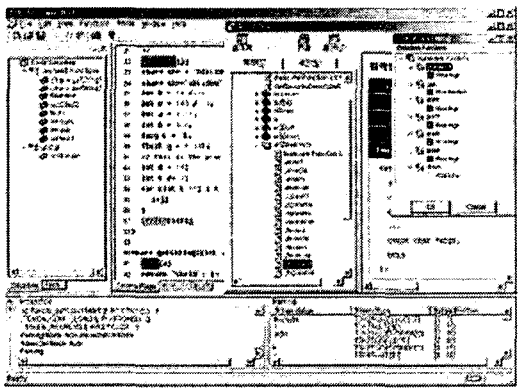
⑤ 각각의 발견된 취약성 함수와 위험성 정보에 대해서 분석 다이얼로그에 나타내고, 분석된

결과에 대해서 별도의 탭에 표시한다.

⑥ 분석 과정이 종료된 후 사용자의 요청에 의해서 각 취약성 함수에 대한 세부적인 해결 방안을 제시해 주는 부가기능을 수행한다.

3.2 구 현

[그림 2]는 파싱과 분석이 끝난 후의 시스템의 전체적인 모습을 보여 준다. 시스템은 소스 코드와 이를 파싱하는 과정과 결과를 나타내는 메인 다이얼로그, 선택한 취약성 함수에 대한 문제점과 이에 대한 해결 방안을 제시하는 취약 함수 다이얼로그, 마지막으로 소스 코드에서 발생한 취약성 함수들의 리스트와 위험도를 나타내는 분석 다이얼로그로 구성된다.



[그림 2] 전체적인 시스템 구성

4. 결 론

취약성을 야기시키는 결점들이 발생하는 위치와 시점은 대부분 프로그램 개발 초기 단계이다. 이런 결점을 원천적으로 보완하기 위해서는 프로그램 작성 단계에서 보다 명확한 보안 개념을 가지고 소스 코드를 작성해야 한다는 것이다.

본 논문에서는 위에 언급한 문제들과 관련하여 표준 C 함수와 Win32 API 함수의 취약성에 관한 대표적인 부류들을 정의하고, 그 원인과 고려해야 할 점에 대하여 기술하였다.

마지막으로 개발 단계에서 공격자들의 목표가 될 수 있는 취약성들, 예를 들어 잠재적인 취약성을 가지는 함수들을 분석하였다. 또한 이에 대한 데이터베이스를 구축하였으며, 자동화된 소스 코드 분석 도구를 설계 및 개발하였다.

참고문헌

[1] A. One, "Smashing the stack for fun and profit," Phrac 7(49), November 1996.

[2] A.baratloo and N.Singh, "Transparent Run-Time Defense Against Stack Smashing Attacks," USENIX Annual Technical Conference, 2000

[3] Mudge, "How to write Buffer Overflows," available at http://www.insecure.org/stf/mudge_buffer_overflow_tutorial.html.

[4] M. Conover, "w00w00 on Heap Overflows," available at http://www.w00w00.org/files/articles/heaptut.txt.

[5] C. Cowan, C. Pu, H. Hinton, J. Walpole, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer - Overflow Attacks," 7 th USENIX Security Conference, 1998.

[6] D. A. Wheeler, "Secure programming for Linux and Unix HOWTO," available at http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html, 2003.

[7] M. Howard and D. LeBlanc, "Writing Secure Code," Microsoft Press, ISBN 0-7356-1588-8, 2002.

[8] T. La, "Secure Software Development and

Code Analysis Tools," available at http://www.sans.org/rr/ paper.php?id=389, 2003.
[9] Pscan, available at http://www.striker.ottawa.on.ca /~aland/pscan/.
[10] Flawfinder, available at http://www.dwheeler.com/ flawfinder.
[11] RATS, available at http://www.securesoftware.com.
[12] Splint, available at http://lclint.cs.virginia.edu/.
[13] MOPS, available at http://www.cs.berkeley.edu/~daw /mops.

하 경 휘

2003년 국민대학교 전산학과(이학사)
현재 국민대학교 전산학과 석사과정

최 진 우

1998년 한성대학교 컴퓨터공학과(공학사)
2000년 국민대학교 전산학과(이학석사)
2004년 국민대학교 전산학과(이학박사)
현재 한국과학기술연구원 POST-DOC
현재 국민대학교 컴퓨터학부 강사

우 종 우

1991년 일리노이 공대 전자계산학과(공학박사)
1994년 ~ 현재 국민대학교 컴퓨터학부 교수

김 홍 철

현재 국가보안기술연구소 정보보증연구부 연구원

박 상 서

현재 국가보안기술연구소 정보보증연구부 선임연구원, 팀장